EML, an Energy Measurement Library

Alberto Cabrera*, Francisco Almeida† and Vicente Blanco†
* Instituto Tecnológico y de Energías Renovables. ITER
Granadilla, Tenerife. Spain
Email: acabrera@iter.es
† HPC Group. ETS de Ingeniería Informática
Universidad de La Laguna, ULL
La Laguna. 38270 Tenerife. Spain
Email: {falmeida, Vicente.Blanco}@ull.es

I. Introduction

Over the past years, energy aware computing has become a hot topic in High Performance Computing (HPC), as the work on the field towards the exascale computing has been progressing. To achieve exascale performance, several aspects of current computational models are being reviewed taking into account the energy consumption.

In the case of the algorithmic analysis, approaches like performance profiling and analytic modeling require good measurement tools to achieve the goals without having to deal with the concrete details of every hardware architecture and their respective challenges, such as how to properly measure energy.

Work in the field has been made to solve the problem of energy measurement by authors like Bellosa [1], Ryffel [2] and Ge et al. [3], but as the work was not directly available or it was a hardware solution, we started to develop a portable library called EML (Energy Measurement Library). With this implementation it is possible to abstract the user from the measuring tools, allowing to perform easiest measurement and, depending on the grade of precision and intrusion that we can have within the experiment, choose between external measurements or code instrumentation, as shown in Figure 1. Uses for this kind of library are decision making and auto-tuning based on energy consumption among other possibilities. To achieve so, we offer an unified interface that has a backend with implementeations for every measurement tool, which is easy to extend.

We currently have for measuring energy consumption the following implementations:

- PDU's
- Intel MSR
- A cluster with a measurement frontend

Figure 2 illustrates an example of measurement within the core using the Intel MSR Interface. It shows ten seconds of three obtained power profiles that belong to an execution of various matrix multiplications using the Intel MKL dgemm routine. Power profiles PP0 and PP1 belong to the processor while DRAM is the DIMM power consumption. Another example of measurement is Figure 3. In this case, measurement corresponds to an execution of High Performance Linpack, and

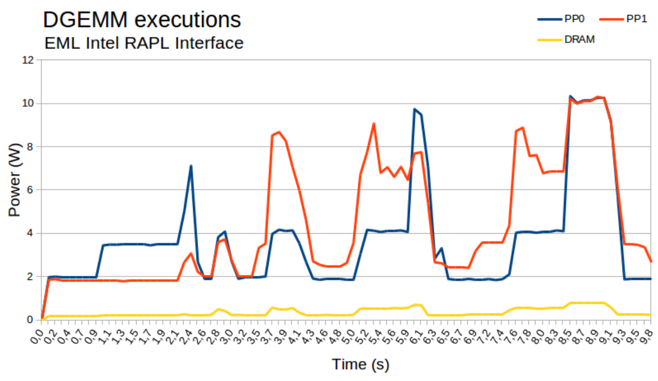


Fig. 2. Energy measurement from a processor using the Intel MSR RAPL interface.

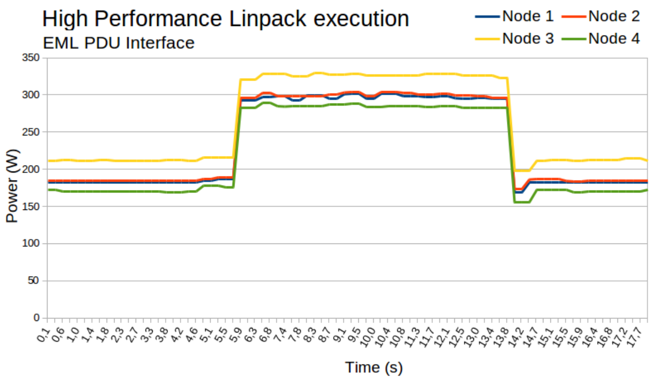
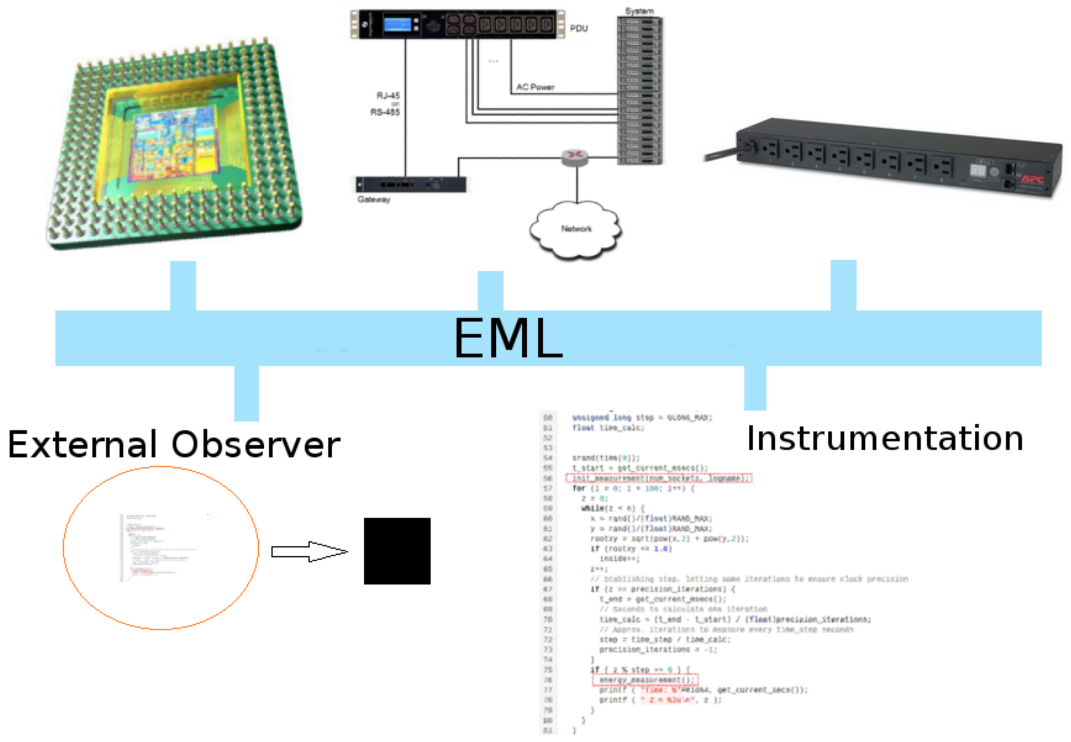


Fig. 3. Energy measurement using an external metered PDU from a monitor node.

is taken directly from pdu outlets using 4 nodes. Both outputs are logged in a file that can be formatted using timestamps to synchronize energy measurement and program execution. EML allows portability for our code in different architectures, provided that the new machine is compatible with at least one of the coded solutions.

II. Details

The current software design is thought to minimize the effort of adding code for new measurement tools. The simplest option is to structure the library as a Factory Method design



```
unsigned long step = ULONG_MAX;
float time_calc;


srand(time(0));
t_start = get_current_msecs();
init_measurement(num_sockets, logname);
for (i = 0; i < 100; i++) {
  z = 0;
  while(z < n) {
    x = rand()/(float)RAND_MAX;
    y = rand()/(float)RAND_MAX;
    rootxy = sqrt(pow(x,2) + pow(y,2));
    if (rootxy <= 1.0)
      inside++;
    z++;
    // Stablishing step, letting some iterations to ensure clock precision
    if (z == precision_iterations) {
      t_end = get_current_msecs();
      // Seconds to calculate one iteration
      time_calc = (t_end - t_start) / (float)precision_iterations;
      // Approx. iterations to measure every time_step seconds
      step = time_step / time_calc;
      precision_iterations = -1;
    }
    if ( z % step == 0 ) {
      energy_measurement();
      printf ( "Time: %"PRId64, get_current_secs());
      printf ( " Z = %lu\n", z );
    }
  }
}
```

Fig. 1. EML overview diagram. The upper part represents different hardware configurations (A processor that has Intel MSR RAPL, an external node to monitor a cluster or direct access to a PDU. The lower part shows the two different ways of adquiring data: an external program that measures energy consumption or code instrumentation. EML acts as a interconnection layer.

pattern, with every measurement tool represented as a class implementing two possible methods:

- *instant_measurement*. A method that returns the instant energy consumption measured by the hardware.
- *interval_measurement*. A method that returns the energy measured in an interval defined by the user. To determine the correct interval, this method is called following the classical time measurement: before and after performing the operation that requires measurement.

This involves a few challenges. First, the measurement tools we have encountered have one of the two methods directly available, so the second method has to be derived from the values obtained from the available one. From *instant_measurement* to *interval_measurement* the operation consists on measuring on the best possible interval given by the hardware constraints and then performing the correspondent integrating operation. The opposite, from *interval_measurement* to *instant_measurement*, is implemented in a similar way. Measurement is done on a given interval, then the instant value in watts is equal to the amount of energy resulting from resolving the equation

$$P = \frac{E}{s} \tag{1}$$

P is our instant power consumption, obtained thanks to the energy E measured every s seconds.

Our second challenge is to unify the different information obtained by the different energy measurement tools that could generate confusion. Given the concrete case of the Intel RAPL MSR and our metered PDU, the first obtains energy for a processor, separated in three different measures: the core itself, the RAM DIMMs and what Intel calls the *uncore* which, in Sandy Bridge platforms, measures the energy of the on–chip graphics processor. The latter returns the energy of a whole node. For now this is resolved by tagging the measurement taken. For future work, we pretend to extend EML capabilities to facilitate the analysis of the data offering different outputs such as XML.

III. Acknowledgments

This work was supported by the Spanish Ministry of Education and Science through the TIN2011-24598 project and the Spanish network CAPAP-H4.

References

[1] F. Bellosa, "The benefits of event: driven energy accounting in power-sensitive systems," in *ACM SIGOPS European Workshop*. ACM, 2000, pp. 37–42.

[2] S. Ryffel, "Lea^{2}p: The linux energy attribution and accounting platform," Master's thesis, Swiss Federal Institute of Technology, 2009.

[3] R. Ge, X. Feng, S. Song, H.-C. Chang, D. Li, and K. W. Cameron, "Powerpack: Energy profiling and analysis of high-performance systems and applications," *IEEE Trans. Parallel Distrib. Syst.*, vol. 21, no. 5, pp. 658–671, 2010.